Not callable from ISR.

Requester will be blocked when the wait option is selected and the memory is not available.

3.7.7 RN_RETSEG

NAME

rn_retseg -- "Return a Segment"

SYNOPSIS

```
#include <memory.h>
uint rn_retseg ( rnid, segaddr )

        uint rnid;      /* region id as returned by rn_create or rn_ident */
        char *segaddr;  /* segment address as returned by rn_getseg */
```

DESCRIPTION

This directive returns a segment to its region. If possible, the segment is merged with neighboring segments. The resulting segment then becomes available for subsequent allocation, or allocation to tasks already waiting.

RETURN VALUE

If *rn_retseg* successfully returned the segment, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *rnid*.

Segment not from specified region.

NOTES

Not callable from ISR.

May cause a preempt if a task waiting for memory becomes ready as a result of this call and has a higher priority than the running task, and the preempt mode is in effect.

3.7.8 PT_CREATE

NAME

pt_create – "Create a Partition"

SYNOPSIS

```
#include <memory.h>
uint pt_create ( name, paddr, length, bsise, flags, &ptid, &bnum)

        uint name;      /* user defined 4-byte partition name */
        char *paddr;    /* physical start address of partition */
        uint length;    /* physical length in bytes */
        uint bsise;     /* size of buffers in bytes */
        uint flags;     /* partition attributes */
        uint ptid;      /* partition id - returned by this call */
        uint bnum;      /* number of buffers in partition - returned by this call */
```

Flags field values:

GLOBAL	set	to indicate the partition is a multiprocessor global resource.
	clear	to indicate the partition is local

DESCRIPTION

This directive allows the user to create a partition of fixed size buffers from a contiguous memory area. The partition id will be returned in *ptid* by the executive to use for *pt_getbuf* and *pt_retbuf* directives for the partition. The number of buffers created by the executive will be returned in *bnum*.

The partition physical start address specified in *paddr* will be long-word aligned by the executive. In systems with an MMU, the partition physical start address must be on the pagesise boundary.

The executive may use memory within the partition for partition and buffer data structures. Therefore, the product of the buffer count and buffer size will be slightly less than the length of the partition.

By setting the GLOBAL value in the flags field, the *ptid* will be sent to all processors in the system, to be entered into a global resource table. The system is defined as the collection of interconnected processors.

The maximum number of partitions that may exist at any one time is a configuration parameter.

RETURN VALUE

If *pt_create* successfully created the partition, the *ptid* and *bnum* are filled in and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Too many partitions.

NOTES

Not callable from ISR.

Will not cause a preempt.

3.7.9 PT_IDENT

NAME

pt_ident -- "Obtain id of a Partition"

SYNOPSIS

```
#include <memory.h>
uint pt_ident ( name, node, &ptid )

        uint name;    /* user defined 4-byte partition name */
        uint node;    /* node identifier */
                      /* 0 indicates any node */
        uint ptid;    /* partition id - returned by this call */
```

DESCRIPTION

This directive allows a task to identify a previously created partition by name and obtain the *ptid* to use for *pt_getbuf* and *pt_retbuf* directives for the partition.

If the partition name is not unique, the *ptid* returned may not correspond to the partition named in this call.

The partition may have been created by the local processor or any remote processor in a multiprocessor configuration, as long as the partition was created with the GLOBAL flags value set (see *pt_create*). If the partition name is not unique within the multiprocessor configuration, a non-zero node identifier must be specified in the *node* field.

RETURN VALUE

If *pt_ident* directive succeeds, the *ptid* will be filled in and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Named partition does not exist.

Invalid node identifier.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

3.7.10 PT_DELETE

NAME

pt_delete – "Delete a Partition"

SYNOPSIS

```
#include <memory.h>
uint pt_delete ( ptid )

        uint ptid;    /* partition id as returned by pt_create or pt_ident */
```

DESCRIPTION

This directive removes a partition, provided that none of its buffers is still allocated.

After this directive has successfully executed, the executive will reject any *pt_getbuf* or *pt_retbuf* directives for the partition.

The partition must exist on the local processor. If the partition was created with the GLOBAL flags value set in a multiprocessor configuration, a notification will be sent to all processors in the system, so the *ptid* can be deleted from the global resource table.

RETURN VALUE

If *pt_delete* successfully removed the partition, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *ptid*.

Cannot delete – some buffers in use.

Partition not created from local node.

NOTES

Not callable from ISR.

Will not cause a preempt.

3.7.11 PT_GETBUF

NAME

pt_getbuf -- "Get a Buffer"

SYNOPSIS

```
#include <memory.h>
uint pt_getbuf ( ptid, &bufaddr )

        uint ptid;          /* partition id as returned by pt_create or pt_ident */
        char *bufaddr;      /* buffer address - returned by this call */
```

DESCRIPTION

The *pt_getbuf* directive will get a buffer from a buffer partition. The buffer address will be returned in *bufaddr* as a result of this call.

The partition may have been created by the local processor or any remote processor in a multiprocessor configuration, as long as the partition was created with the **GLOBAL** flags value set (see *pt_create*).

RETURN VALUE

If *pt_getbuf* successfully got a buffer, then the address of the buffer is returned in *bufaddr* and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *ptid*.

Partition out of free buffers.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

3.7.12 PT_RETBUF

NAME

pt_retbuf – "Return a Buffer"

SYNOPSIS

```
#include <memory.h>
uint pt_retbuf ( ptid, bufaddr )

        uint ptid;          /* partition id as returned by pt_create or pt_ident */
        char *bufaddr;      /* buffer start address as returned by pt_getbuf */
```

DESCRIPTION

The *pt_retbuf* directive will return a buffer to the partition from which it was originally allocated.

Buffers are not automatically released when a task is deleted.

RETURN VALUE

If *pt_retbuf* successfully returned the buffer, then 0 is returned.

If the buffer was not returned, a value of -1 is returned, and *errno* is set to indicate the error.

ERROR CONDITIONS

Invalid *ptid*.

Buffer not from specified partition.

NOTES

Can be called from within an ISR.

Will not cause a preempt.

3.8 MMU Management

The executive can optionally support the PMMU (M68851 and M68030) to provide memory protection, dynamic task loading, and dynamic memory allocation.

To provide these services, the executive adopts an MMU model which defines the *pagesize*, the structure and depth of the memory map tree, and the degree of control each task has over its own memory map. Different implementations of the RTEID are free to choose different models. However, the model chosen should allow the standard memory management services (regions and partitions) to operate in a consistent and intuitive manner in both an MMU and non-MMU environment.

Logically, the RTEID adopts a sectioned view of the logical address space associated with each task. Memory objects are mapped into a task's logical address space in variable size MMU sections. A single section is contiguous in the logical and possibly the underlying physical address spaces. Thus, the MMU is used to define a set of mappings for each task in the form:

(logical address, length) --> physical address range

Based on this model, the RTEID defines how the memory management services should operate, and defines additional services to manage the MMU directly.

3.8.1 Segments vs. Sections

MMU sections should not be confused with region segments. A segment is a block of memory allocated from a region. It can exist on any CPU in the M68000 family. A section is only meaningful on the M68030 or M68020/M68851 combination, and refers to a contiguous block of memory which is mapped into a task's address space.

3.8.2 Regions

When a task calls *rn_getseg* to obtain a segment from a region, the segment is automatically mapped into the task's logical address space at an executive assigned address. Because *rn_getseg* performs the mapping, the corresponding region is not mapped into the address space of tasks using it. This means that allocated sections are accessible only by the allocating task, and those tasks which explicitly are given access to the segment using the MMU directives. Thus, a segment is fully protected from inadvertent access by other tasks.

3.8.3 Partitions

When a task executes a *pt_create* or *pt_ident* directive, the entire partition is mapped into the task's address space. Thus, tasks which share a partition can share and access any buffers allocated from the partition. However, protection is on the partition level, and individual buffers are not protected.

The directives provided by the MMU manager are:

Directive	Function
mm_l2p	Logical to physical
mm_p2l	Physical to logical
mm_pmap	Map physical
mm_unmap	Unmap logical
mm_pread	Physical read
mm_pwrite	Physical write
mm_ptcreate	Create logical partition

3.8.4 MM_L2P

NAME

mm_l2p -- "Logical to Physical"

SYNOPSIS

```
#include <memory.h>
uint mm_l2p ( tid, laddr, &paddr, &length )

        uint tid;       /* task id as returned by t_create or t_ident */
        char *laddr;    /* logical start address */
        char *paddr;    /* physical start address - returned by this call */ */
        uint length;    /* remaining length in bytes - returned by this call */
```

DESCRIPTION

This directive calculates the physical address within the section associated with the logical address belonging to the task identified by the *tid*.

The physical start address is returned in the *paddr* field. The number of bytes remaining in the section is returned in the *length* field.

RETURN VALUE

If *mm_l2p* was successful, then the physical start address is returned in *paddr*, the number of bytes remaining is returned in *length*, and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Unmapped logical address.

Task not created on local node.

ISR cannot reference remote node.

NOTES

Can be called from within an ISR, except when the task was not created on the local node.

Will not cause a preempt.

3.8.5 MM_P2L

NAME

mm_p2l – "Physical to Logical"

SYNOPSIS

```
#include <memory.h>
uint mm_p2l ( tid, paddr, &laddr, &length )

        uint tid;       /* task id as returned by t_create or t_ident */
        char *paddr;    /* physical start address */
        char *laddr;    /* logical start address - returned by this call */ */
        uint length;    /* remaining length in bytes - returned by this call */
```

DESCRIPTION

This directive returns the logical address within the section associated with the physical address belonging to the task identified by the *tid*. The executive will only return the first valid mapping of the physical address it finds, and the logical address returned may be ambiguous if the task has a many-to-one mapping of the physical address range.

The logical start address is returned in the *laddr* field, and the number of bytes remaining in the section is returned in the *length* field.

RETURN VALUE

If *mm_p2l* was successful, then the logical address is returned in *laddr*, the number of bytes remaining is returned in *length*, and 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Unmapped logical address.

Task not created on local node.

NOTES

Not callable from ISR.

Will not cause a preempt.

3.8.6 MM_PMAP

NAME

mm_pmap -- "Map Physical"

SYNOPSIS

```
#include <memory.h>
uint mm_pmap ( tid, laddr, paddr, length, flags )

        uint tid;       /* task id as returned by t_create or t_ident */
        char *laddr;    /* logical start address */
        char *paddr;    /* physical start address */
        uint length;    /* length in bytes */
        uint flags;     /* section attributes */
```

The flags field values are defined as follows:

RDONLY	set	read-only
	clear	read-write

DESCRIPTION

This directive maps physical memory starting at *paddr* for the number of bytes specified in *length,* to a section at the logical start address *laddr* in the address space of the task identified by the *tid.*

The physical start address specified in *paddr* must be on the pagesize boundary. The logical start address specified in *laddr* must be on a section boundary.

If *length* is not a multiple of the pagesize, then more bytes than requested are mapped.

RETURN VALUE

If *mm_pmap* was successful, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid.*

Paddr is not on a pagesize boundary.

Laddr is not on a section boundary.

Length specified is too large.

Duplicate logical address.

Task not created on local node.

ISR cannot reference remote node.

NOTES

Can be called from within an ISR, except when the task was not created on the local node.

Will not cause a preempt.

3.8.7 MM_UNMAP

NAME

mm_unmap -- "Unmap Logical"

SYNOPSIS

```
#include <memory.h>
uint mm_unmap ( tid, laddr )

        uint tid;       /* task id as returned by t_create or t_ident */
        char *laddr;    /* logical start address */
```

DESCRIPTION

This directive removes the section starting at logical address *laddr* from the address space of the task identified by the *tid*.

RETURN VALUE

If *mm_unmap* was successful, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

Invalid *tid*.

Unmapped logical address.

Task not created on local node.

ISR cannot reference remote node.

NOTES

Can be called from within an ISR, except when the task was not created on the local node.

Will not cause a preempt.

To return the segment to the region, the *rn_retseg* directive must be used.

3.8.8 MM_PREAD

NAME

mm_pread – "Physical read"

SYNOPSIS

```
#include <memory.h>
uint mm_pread ( paddr, laddr, length )

        uint paddr;     /* physical start address */
        char *laddr;    /* logical start address */
        uint length;    /* length in bytes */
```

DESCRIPTION

The *mm_pread* directive reads from a physical address, and writes to the logical address in the calling task's address space. The length cannot span a section boundary.

RETURN VALUE

If *mm_pread* was successful, then 0 is returned.

If the call was not successful, no data is transferred and an error code is returned.

ERROR CONDITIONS

Unmapped logical address.

Length spans section boundary.

NOTES

Not callable from ISR.

Will not cause a preempt.